

Q.1.

; Write an assembly language program to display a pixel on the screen.

```
dosseg
.model small
.stack 100h

.data

.code

main proc

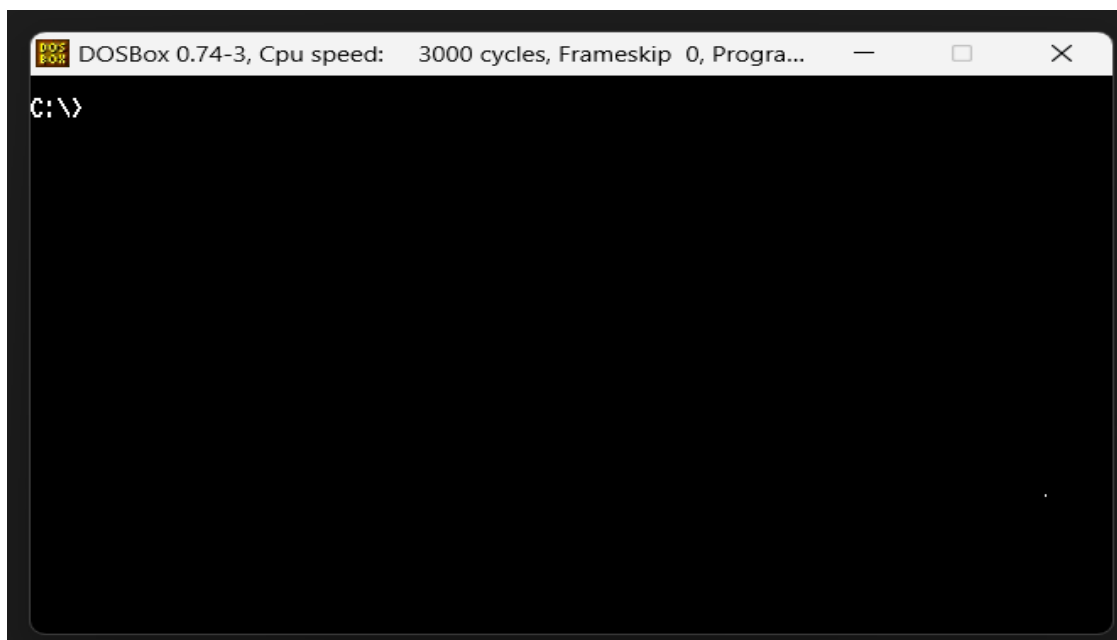
; setting the graphic mode

mov ah,0h ; video mode is being set
mov al,6h
int 10h

; displaying the pixel
mov ah,0ch
mov al,1h ; white color ky liyay
mov cx,600 ; col
mov dx,150 ; row
int 10h

mov ah,4ch
int 21h

main endp
end main
```



Q.2.

;Write an assembly language code to draw a line on the screen.

```
dosseg
```

```
.model small
```

```
.stack 100h
```

```
.data
```

```
.code
```

```
main proc
```

```
mov ah,6
```

```
mov al,1
```

```
mov bh,00010000b
```

```
mov ch,0
```

```
mov cl,5
```

```
mov dh,10
```

```
mov dl,60
```

```
int 10h
```

```
mov ah,4ch
```

```
int 21h
```

```
main endp
```

```
end main
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Drive :
Z:\>C
    1 hellonew.asm
C:\>msoft (R) Macro Assembler Version 6.11
Microsoft (C) Microsoft Corp 1981-1993. All rights reserved.
Copyr
    mbling: hellonew.asm
Asse
Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.

Object Modules [l.obj]: hellonew.obj
Run File [hellonew.exe]: "hellonew.exe"
List File [nul.map]: NUL
Libraries [l.lib]:
Definitions File [nul.def]:

C:\>hellonew.asm
Illegal command: hellonew.asm.

C:\>hellonew.exe

C:\>_
```

Q.3.

;Write an assembly language code to draw a square on the screen.

```
dosseg
.model small
.stack 100h

.data

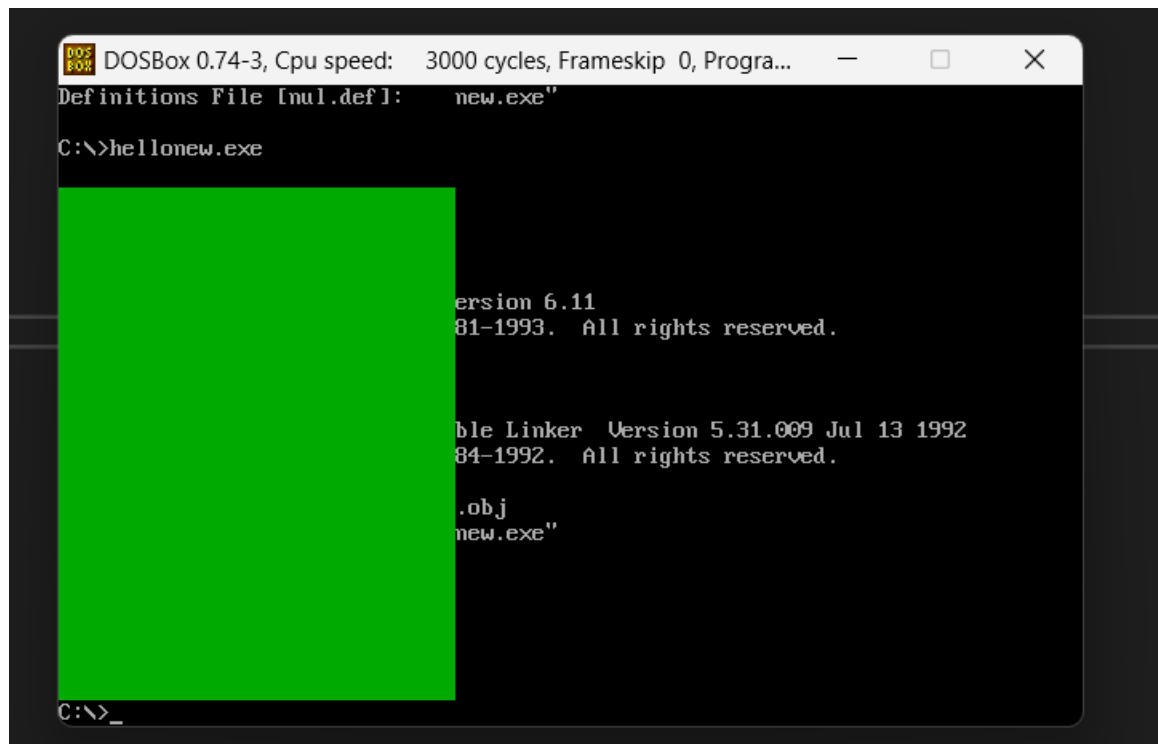
.code

main proc

mov ah,6
mov al,20
mov bh,00100000b
mov ch,0
mov cl,0
mov dh,30
mov dl,30
int 10h

mov ah,4ch
int 21h

main endp
end main
```



Q.4.

```
.model small
.stack 100h
.data
    x dw 100
    y dw 120
    temp dw ?
.code
main proc
    mov ax,@data
    mov ds,ax
    mov ah,0
    mov al,6
    int 10h

    mov cx,100

horizontal_line:

    mov temp,cx
    mov ah,0ch
    mov al,0Dh
    mov cx,x
    mov dx,y
    inc x
    int 10h
    mov cx,temp

loop horizontal_line

    mov cx,50
```

```

left_line:
    mov temp,cx
    mov ah,0ch
    mov al,0Dh
    mov cx,x
    mov dx,y
    dec x
    dec y    ; we decrement here instead of increment as dec causes moving
upwards while inc cause y to move downwards , kind of like opposite
    int 10h
    mov cx,temp

loop left_line

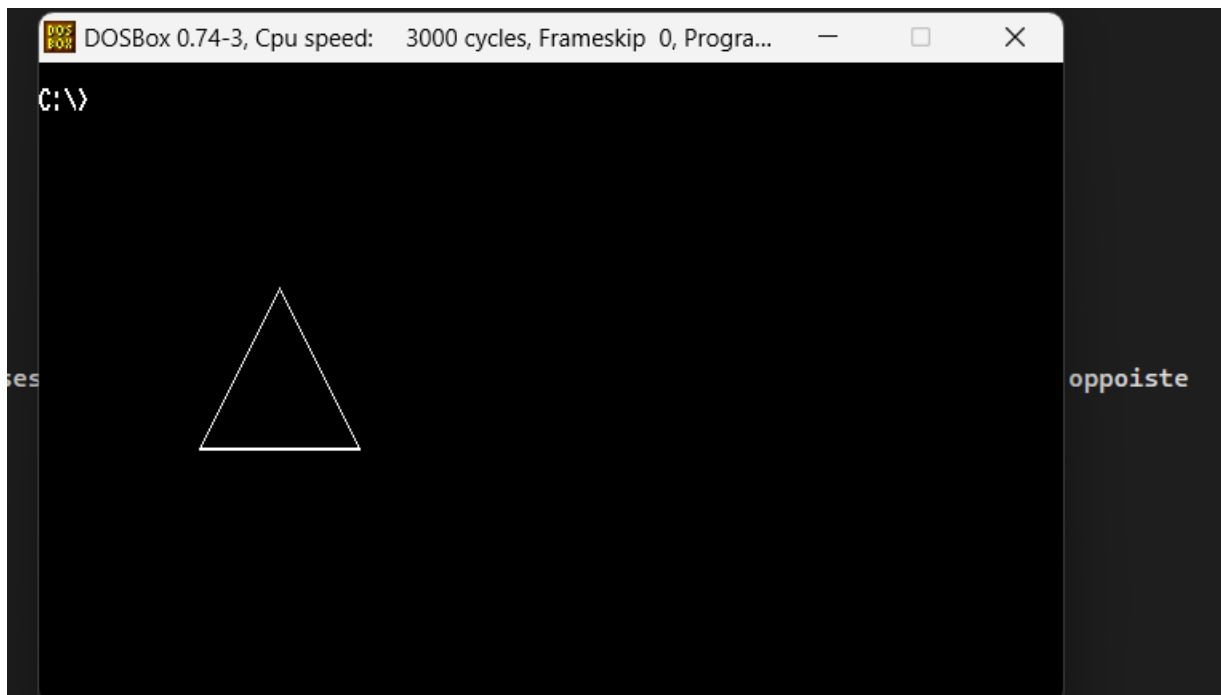
    mov cx,50

rightline:
    mov temp,cx
    mov ah,0ch
    mov al,0Dh
    mov cx,x
    mov dx,y
    dec x
    inc y
    int 10h
    mov cx,temp

loop rightline

    mov ah, 4ch
    int 21h
main endp
end main

```



```
.model small
.stack 100h
.data
    x dw 100
    y dw 120
    temp dw ?
.code
main proc
    mov ax,@data
    mov ds,ax
    mov ah,0h
    mov al,6h
    int 10h

    mov cx,100

horizontal_line:
    mov temp,cx
    mov ah,0ch
    mov al,0Dh
    mov cx,x
    mov dx,y
    inc x
    int 10h
    mov cx,temp
loop horizontal_line

    mov cx,50

rightline:
    mov temp,cx
    mov ah,0ch
```

```

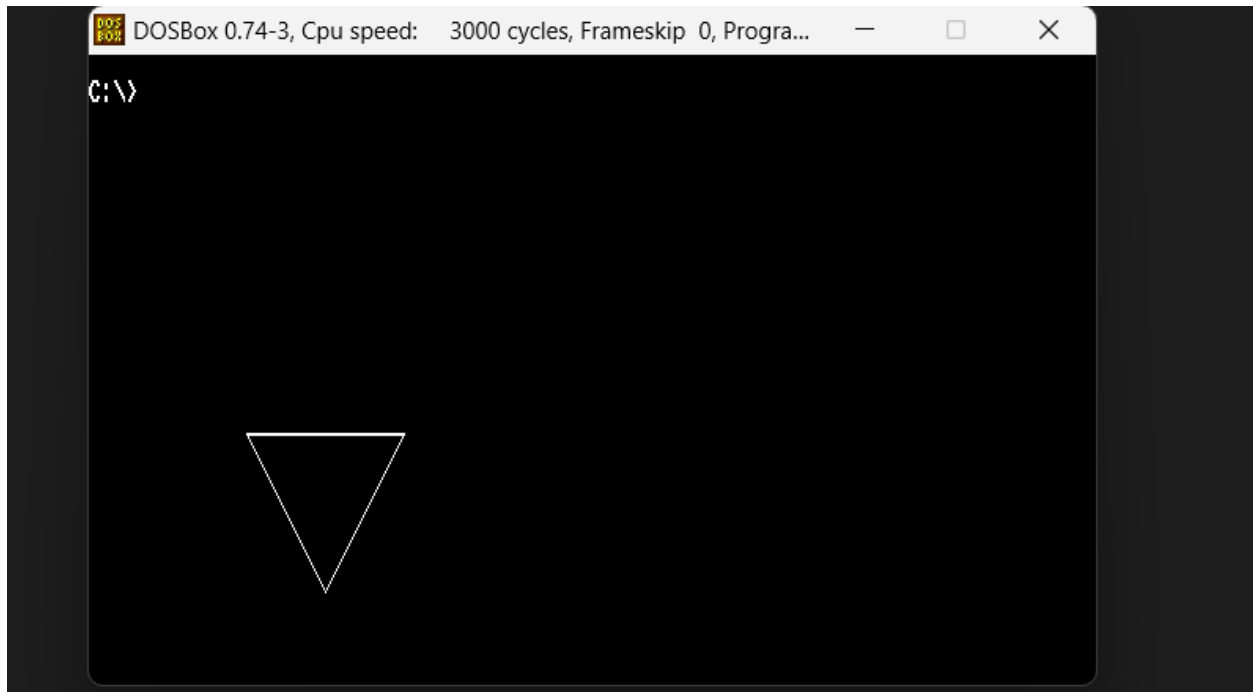
        mov al,0Dh
        mov cx,x
        mov dx,y
        dec x
        inc y
        int 10h
        mov cx,temp
    loop rightline

    mov cx,50

left_line:
    mov temp,cx
    mov ah,0ch
    mov al,0Dh
    mov cx,x
    mov dx,y
    dec x
    dec y    ; we decrement here instead of increment as dec causes moving
upwards while inc causes y to move downwards , kind of like opposite
    int 10h
    mov cx,temp
    loop left_line

    mov ah, 4ch
    int 21h
main endp
end main

```



Q.5.

```
.model small
.stack 100h
.data
    x dw 50
    y dw 50
    r dw 10

    center_y dw 100
    center_x dw 100

    rightSide dw ?
    leftSide dw ?

    x_m dw ?
    y_m dw ?

.code
main proc
    mov ax,@data
    mov ds,ax
    mov ah,0h
    mov al,6h
    int 10h

    mov cx,360

    ;(x-r)^2 + (y-r)^2 <= r^2

l1:
    push cx

    mov ax,r
    mov bx,r
    mul bx
    mov rightSide,ax

    mov ax,x
    sub ax,r

    mov bx,ax
    mul bx
    mov x_m,ax

    mov ax,y
    sub ax,r

    mov bx,ax
    mul bx
    mov y_m,ax

    mov ax,x_m
    add ax,y_m
```



```
    mov leftSide,ax

    mov ax,leftSide

    cmp ax,rightSide

    jle here

    jmp overhere

here:
    mov ah,0ch
    mov al,0fh
    mov cx,x
    mov dx,y
    inc x
    int 10h

overhere:
    inc x
    inc y
    pop cx

loop l1

    mov ah,4ch
    int 21h
main endp
end main
```

